PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-114516
(43)Date of publication of application : 02.05.1995

(51)Int.Cl. G06F 15/16
G06F 9/45

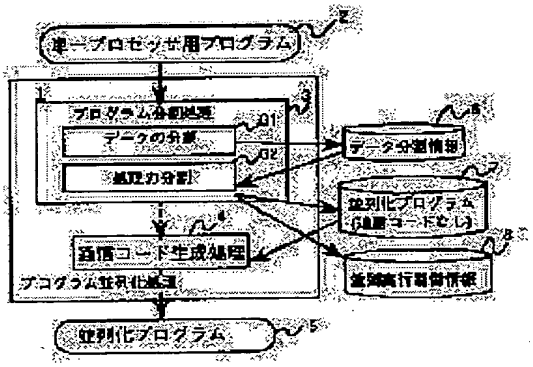
(21)Application number : 06-153273
(22)Date of filing : 05.07.1994

(71)Applicant : HITACHI LTD
HITACHI VLSI ENG CORP
(72)Inventor : OKOCHI TOSHIO
KONNO CHISATO
IKAI MITSUYOSHI

(30)Priority
Priority number : 05191922 Priority date : 06.07.1993 Priority country : JP

(54) PROGRAM PARALLELIZING METHOD

(57)Abstract:
PURPOSE: To reduce the inter-processor communication of a program for parallel computers in a process for converting a program, described in a high-level language, into the program for the parallel computers.
CONSTITUTION: The variables of the inputted program are divided and assigned to plural processors. The inputted program 2 is analyzed, the presence of reference to a variable value defined by other processors is detected as to each process, and codes of a program part regarding the inter-processor communication for the reference are generated. When each processor executes the parallelized program 5, the codes of a program part for deciding whether or not reference to the variable value is already performed are generated by an analysis. A program part which performs control so as to perform the inter- processor communication for executing the reference to the variable value when the reference is not performed, or not to perform the reference when the reference is already done, is generated.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平7－114516

(43)公開日　平成7年（1995）5月2日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　15/16	４３０　Ａ	9229－5B	Ｇ０６Ｆ　9/44	３２２　Ｆ	
9/45					

審査請求　未請求　請求項の数13　ＯＬ（全 14 頁）

(21)出願番号　特願平6－153273

(22)出願日　平成6年（1994）7月5日

(31)優先権主張番号　特願平５－１９１９２２

(32)優先日　平5（1993）7月6日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目6番地

(71)出願人　000233468
日立超エル・エス・アイ・エンジニアリング株式会社
東京都小平市上水本町5丁目20番1号

(72)発明者　大河内　俊夫
東京都国分寺市東恋ケ窪1丁目280番地　株式会社日立製作所中央研究所内

(74)代理人　弁理士　小川　勝男

最終頁に続く

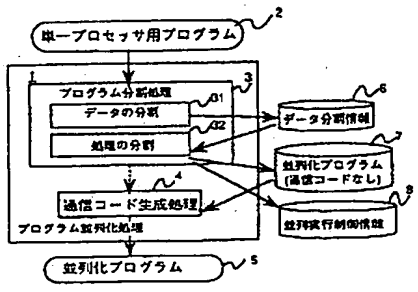
(54)【発明の名称】プログラム並列化方法

(57)【要約】　(修正有)

【目的】　高級言語で記述されたプログラムを並列計算機用プログラムに変換する処理において、並列計算機用プログラムのプロセッサ間通信を削減する。

【構成】　入力されたプログラムの変数は、分割されて複数のプロセッサに割り当てられる。入力されたプログラム２を解析して、各プロセッサについて、他のプロセッサで定義された変数値への参照の存在を検出し、その参照のためのプロセッサ間通信に関するプログラム部分のコードを生成する。また、解析により、各プロセッサについて、そのプロセッサが並列化プログラムを実行時に、その変数値への参照が実行済みかどうかを判定するプログラム部分のコードを生成する。さらに、まだ実行されていない場合には、その変数値への参照を実行するためにプロセッサ間通信をおこなうように制御し、参照済みである場合には、プロセッサ間通信をおこなわないように制御するプログラム部分を生成する。

(図１)



1

【特許請求の範囲】

【請求項1】単一プロセッサ用のプログラムを入力し、該入力されたプログラムを並列プロセッサ用の並列化プログラムに計算機によって変換するプログラム並列化方法であって、

前記入力されたプログラムを解析することにより、各プロセッサについて、変換された並列化プログラムを当該プロセッサによって実行した場合に必要となる可能性のあるプロセッサ間通信を検出し、該通信のためのコードを生成する第1のステップと、

前記入力されたプログラムを解析することにより、各プロセッサについて、当該プロセッサによって前記並列化プログラムを実行した際に、他のプロセッサとの間の通信の要否を判定して通信が必要であると判定された場合にのみ通信を行うように制御するためのコードを生成する第2のステップと、

前記入力されたプログラムに、前記通信のためのコードと前記制御のためのコードとを付加した並列化プログラムを生成する第3のステップとを含むもの。

【請求項2】前記第2のステップは、前記プロセッサ間通信を行うことが常に必要であるプログラム部分には、前記第3のステップにおける前記制御のためのコードを生成せず、無条件に前記通信のためのコードによる通信を行うようにすることを特徴とする請求項1記載のプログラム並列化方法。

【請求項3】前記第2のステップは、前記第1のステップで検出した複数プロセッサにより並列に実行した場合に必要となる可能性のあるプロセッサ間通信について、前記通信が必要となる条件が成立したか否かを管理するためのフラグを新たに設け、該フラグの値を更新するコードと、該フラグの値によって通信の実行を制御するコードとを生成するステップを含むことを特徴とする請求項1記載のプログラム並列化方法。

【請求項4】前記第1のステップが、

前記入力プログラムから、その入力プログラムの実行の流れを有向グラフで表現した制御フローグラフを作成する制御フロー解析ステップと、

該制御フローグラフから、前記入力プログラム中の変数の定義・参照の関係を有向グラフで表現したデータ依存グラフを作成するデータ依存解析ステップと、

該データ依存グラフ中のそれぞれのデータの定義・参照の関係について、プロセッサ間で通信が必要なものを選びだし、通信対象データ依存グラフを作成する通信データ抽出処理ステップと、

該通信対象データ依存グラフからプログラム中で通信を行う箇所を決定し、その位置に挿入すべき通信のためのコードを生成・挿入して通信情報リストを作成する通信位置決定処理ステップとを含むことを特徴とする請求項1記載のプログラム並列化方法。

【請求項5】前記第2のステップは、

2

前記通信情報リスト中の通信コードの通信対象データが定義されているところで所定のフラグをセットするコードを生成する変数定義フラグ生成処理ステップと、

該フラグがセットされているときにのみ通信コードが実行されるように制御するための通信条件式のコードを生成する通信条件式生成処理ステップとを含むことを特徴とする請求項4記載のプログラム並列化方法。

【請求項6】前記第2のステップは、さらに、前記通信の実行後に前記フラグをリセットするコードを生成する変数定義フラグ初期化情報生成処理ステップを含むことを特徴とする請求項5記載のプログラム並列化方法。

【請求項7】前記第2のステップは、さらに、前記通信条件式で参照するフラグについて、前記制御フローグラフを逆に辿ってそのフラグを定義する式を調べ、前記通信条件式が常に成立する場合は、その通信条件式およびフラグを削除する通信条件簡約化処理ステップを備えたことを特徴とする請求項6記載のプログラム並列化方法。

【請求項8】単一プロセッサ用のプログラムを入力し、該入力されたプログラムを並列プロセッサ用の並列化プログラムに計算機によって変換する方法であって、

(a) 前記入力されたプログラムの変数を分割して、複数のプロセッサに割り当てるステップと、

(b) 前記入力されたプログラムの処理を分割して、前記複数のプロセッサに割り当てるステップと、

(c) 前記入力されたプログラムを解析することにより、各プロセッサについて、他のプロセッサで定義された変数値への参照の存在／不存在を検出し、該参照のためのプロセッサ間通信に関するプログラム部分のコードを生成するステップと、

(d) 前記入力されたプログラムを解析することにより、各プロセッサについて、当該プロセッサが並列化プログラムを実行時に、前記変数値への参照が既に実行済みかどうかの判定に関するプログラム部分のコードを生成するステップと、

(e) 各プロセッサについて、当該プロセッサが並列化プログラムを実行時に、前記変数値への参照がまだ実行されていない場合には、前記変数値への参照を実行するためにプロセッサ間通信をおこなうように制御し、既に参照済みである場合には、プロセッサ間通信をおこなわないように制御するプログラム部分を生成するステップとを含むことを特徴とするプログラム並列化方法。

【請求項9】単一プロセッサ用のプログラムを、並列プロセッサで実行されるプログラムに計算機により変換する方法であって、

各々のプロセッサが参照しかつ他のプロセッサによって更新されるデータについて、該プロセッサが最新の値を取得済みであるか否かを示す変数を設け、

該データを更新するプログラム部分で該変数を通信未実行を示す値にする実行文列と、該データの通信を行うプ

ログラム部分で通信実行済みを示す値にする実行文列と、該データを参照するプログラム部分で該制御変数の値を検査し、
該データの通信が済んでいない場合にはプロセッサ間通信によって該データの最新の値を取得する実行文列とを生成することを特徴とするプログラム並列化方法。

【請求項１０】各プロセッサが参照し、他のプロセッサによって更新される変数値を、実行中常にまとまって参照される変数値群の直和に分割し、
該変数値群毎に制御変数を設け、
該変数値群のプロセッサ間通信を行ったときに該制御変数を通信実行済みを示す値にする実行文列を生成することを特徴とする請求項９記載のプログラム並列化方法。

【請求項１１】単一プロセッサ用のプログラムを、並列プロセッサで実行されるプログラムに変換する方法であって、
複数のプロセッサに分散された配列データのうち各々のプロセッサが参照しかつ他のプロセッサによって更新される部分について、該プロセッサが最新の値を取得済みであるか否かを示す制御変数を設け、
該配列を更新するプログラム部分で該変数を通信未実行を示す値にする実行文列と、該データの通信を行うプログラム部分で通信実行済みを示す値にする実行文列と、該データを参照するプログラム部分で該制御変数の値を検査し、該データの通信が済んでいない場合にはプロセッサ間通信によって該データの最新の値を取得する実行文列を生成することを特徴とするプログラム並列化方法。

【請求項１２】該配列データの各プロセッサが参照しかつ他のプロセッサによって更新される部分を、実行中常にまとまって参照される要素群の直和に分割し、
該要素群毎に制御変数を設け、該配列データの要素群の通信を行ったときに該制御変数を通信実行済みを示す値にする実行文列を生成することを特徴とする請求項１０記載のプログラム並列化方法。

【請求項１３】該配列データの各プロセッサが参照しかつ他のプロセッサによって更新される部分を、実行中常にまとまって参照される要素群の直和に分割し、
該配列に対応して該配列データの各要素群が通信済みであるか否かを管理する制御変数を設け、
該配列データの要素群の通信を行うプログラム部分で、該制御変数を該要素群が通信済みであることを示す値にする実行文列と、該要素群を参照するプログラム部分で、該制御変数の値を検査し該要素群の通信が済んでいない場合にはプロセッサ間通信によって該要素群の最新の値を取得する実行文列とを生成することを特徴とする請求項１１記載のプログラム並列化方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、高級言語で記述された

単一プロセッサ用プログラムを分散メモリ型並列プロセッサ用のプログラムに自動変換する技術に関し、特に、並列プロセッサ用のプログラムに必要なプロセッサ間の通信を行うプログラム部分を自動的に計算機により生成する技術に関する。

【０００２】

【従来の技術】プログラムの並列化をおこなうためには、すなわち、ＦＯＲＴＲＡＮなどの高級言語で記述された単一プロセッサ用プログラムから分散メモリ型並列プロセッサ用のプログラム（以下「並列化プログラム」と呼ぶ）を生成するためには、通信を行うプログラム部分を付加することが必要である。　これは、複数のプロセッサで構成される分散メモリ型並列プロセッサシステムにおいて、例えば、変数は各プロセッサに分散して割り当てられるため、あるプロセッサが他のプロセッサで定義された変数を参照する場合に、その参照すべき変数値をプロセッサ間通信によって獲得する必要があるためである。

【０００３】このため、ＦＯＲＴＲＡＮなどの高級言語で記述されたプログラムを入力し並列化プログラムに自動的に計算機によって変換する際には、計算機は、プロセッサ間のデータの通信を行うプログラム部分のコードを自動的に生成し、並列化プログラム中に付与することが行われている。　また、生成された並列化プログラムは、各プロセッサに対するプログラム部分を比較すれば、ほぼ同一の記述であることが望ましい。

【０００４】このように、高級言語で記述されたプログラムから、プロセッサ間のデータの通信を行うコードを含む並列化プログラムを自動生成する技術に関しては、例えば「Proceedings of the 1992 ACM International Conference on Supercomputing」に記載されている。これは、高級言語で記述されたプログラムを計算機が解析して、どのタイミングでどのデータを通信すべきかを検出し、並列化プログラムに自動的に通信コードを挿入するものである。

【０００５】また、物理現象を高水準の問題記述言語で記述したプログラムを計算機が入力し、プロセッサ間通信を記述した並列プロセッサ用の並列化プログラムに変換する技術に関しては、例えば JP—A—4—190422 に記載されている。

【０００６】

【発明が解決しようとする課題】上述の従来技術では、計算機は並列化プログラムの生成時に、各プロセッサが、生成された並列化プログラムを実際に実行しなくても分かる範囲の静的な状況のみを考慮して、通信コードを生成し付加している。しかし、並列化プログラムを実際に動作させたときには、種々の動的な状況の変化があるので、ある位置におけるプロセッサ間通信の要否が並列化プログラムの実行時にしか決まらない場合もある。したがって、前記従来技術によれば、必要のない無駄な

通信を行なう可能性があった。

【0007】生成された並列化プログラムの実行性能を高めるためには、プロセッサ間の通信を必要最小限にとどめること、すなわち、無駄なプロセッサ間通信を行わないようにするのが望ましい。前記従来技術では、通信の要否が並列化プログラムの実行時にしか決まらないもの（例えば、元のプログラム中に条件文があって、その条件に応じてプロセッサ間通信の要否が決まるものなど）について、無駄な通信を行わないようにする方法については触れられていない。

【0008】本発明の目的は、プロセッサ間通信の要否が実行時にしか決まらない場合についても、並列化プログラムを各プロセッサが実行する時に、無駄なプロセッサ間通信が生じないような並列化プログラムを単一プロセッサ用プログラムから自動的に変換生成することができる方法を提供することにある。

【0009】

【課題を解決するための手段】前記目的を達成するため、本発明は、単一プロセッサ用のプログラムを入力し、該入力されたプログラムを並列プロセッサ用の並列化プログラムに計算機によって変換する方法であって、前記入力されたプログラムを解析することにより、各プロセッサについて、変換された並列化プログラムを当該プロセッサによって実行した場合に必要となる可能性のあるプロセッサ間通信を検出し、該通信のためのコードを生成する第１のステップと、前記入力されたプログラムを解析することにより、各プロセッサについて、当該プロセッサによって前記並列化プログラムを実行した際に、他のプロセッサとの間の通信の要否を判定して通信が必要であると判定された場合にのみ通信を行うように制御するためのコードを生成する第２のステップと、前記入力されたプログラムに、前記通信のためのコードと前記制御のためのコードとを付加した並列化プログラムを生成する第３のステップとを含むことを特徴とする。

【0010】なお、プロセッサ間通信を行うことが常に必要であると計算機が判定したプログラム部分には、前記第３のステップによる制御のためのコードを生成せず、無条件にプロセッサ間通信を行うようにするとよい。

【0011】また、前記変数値への参照が当該プロセッサによって既に実行済みか否かを管理するためのフラグを新たに設け、該フラグの値を更新するコードと、該フラグの値に応じてプロセッサ間通信の実行を制御するコードとを生成するようにするとよい。

【0012】例えば、前記第１のステップは、前記入力されたプログラムから、その入力されたプログラムの実行の流れを有向グラフ（directed graph）で表現した制御フローグラフを作成する制御フロー解析ステップと、該制御フローグラフから、前記入力されたプログラム中の変数の定義・参照の関係を有向グラフで表現したデータ依存グラフを作成するデータ依存解析ステップと、該データ依存グラフ中のそれぞれのデータの定義・参照の関係について、プロセッサ間で通信が必要なものを選びだし、通信対象データ依存グラフを作成する通信データ抽出処理ステップと、該通信対象データ依存グラフからプログラム中で通信を行う箇所を決定し、その位置に挿入すべき通信のためのコードを生成・挿入して通信情報リストを作成する通信位置決定処理ステップとを備えるようにしてもよい。

【0013】また、前記第２のステップは、前記通信情報リスト中の通信コードの通信対象データが定義されているところで所定のフラグをセットするコードを生成する変数定義フラグ生成処理ステップと、該フラグがセットされているときにのみ通信コードが実行されるように制御するための通信条件式のコードを生成する通信条件式生成処理ステップとを備えてもよい。

【0014】さらに、前記通信の実行後に前記フラグをリセットするコードを生成する変数定義フラグ初期化情報生成処理ステップ、あるいは前記通信条件式で参照するフラグについて、前記制御フローグラフを逆に辿ってそのフラグを定義する式を調べ、前記通信条件式が常に成立する場合は、その通信条件式およびフラグを削除する通信条件簡約化処理ステップを備えてもよい。

【0015】

【作用】本発明のプログラム変換方法によれば、プロセッサ間通信の要否が実行時にしか決まらないような場合にも、プロセッサ間通信の要否を実行時に判断して選択的に通信を行うことにより、無駄な通信を行わないような並列化プログラムを計算機によって自動生成することができる。

【0016】

【実施例】以下、図面を参照して本発明の一実施例を説明する。

【0017】図１は、本発明に係るプログラム並列化方法を実現する計算機システムの実施例の概略構成および概略処理を示す図である。図２は、図１に示す通信コード生成処理４の詳細な手順を示すチャートである。これらの手順については、後に詳述する。

【0018】図１に示すように、計算機１は、元の単一プロセッサ用のプログラム２を入力し、入力されたプログラム２に対して、複数のプロセッサに分割する処理３および各プロセッサについて通信コードを生成する処理４を含む本実施例のプログラム並列化処理を実行し、生成された並列化プログラム５を出力する。プログラム分割処理３は、データを各プロセッサに分割するデータ分割処理３１および処理を各プロセッサに分割する処理の分割処理３２を含んでいる。計算機１は、処理の過程で生成されるデータ分割情報６、通信コードなしの並列化プログラム７および並列実行用制御情報８をそれぞれファイルに格納する。

【００１９】図３は、単一プロセッサ用のプログラムの例を示す。図４は、図３に示した単一プロセッサ用のプログラムを従来のプログラム並列化方法により並列化した場合のプログラムの例である。また、図５は、図３に示した単一プロセッサ用のプログラムを本発明のプログラム並列化方法により並列化したプログラムの例である。図１０、図１２および図１４は、図３に示した単一プロセッサ用のプログラムを本発明の実施例により並列化する処理における、通信コード生成処理４の中間段階のデータを示す。

【００２０】以下、まず図３に示す単一プロセッサ用の元のプログラム２００を例にして、本実施例によって並列化したプログラム５０１（図５）と、従来方式によって並列化したプログラム５００（図４）とを、比較しながら説明する。

【００２１】図３の元のプログラム２００において、右側に付した（１０），（１１），（２０）、…などは文番号を示す。この元のプログラム２００では、まず配列インデックスＩが１から１００の範囲について、配列Ｂ（Ｉ）の各要素の値をすべて０．０に初期化している（文１０，１１）。次に、文２１から文２９までの処理を、変数ＴＩＭＥに設定された回数だけ、繰り返している（文２０，３０）。

【００２２】文２１～２４では、配列要素Ｂ（１）の値がＩ．０を越えるとき、配列インデックスＩが１から１００の範囲について配列Ａ（Ｉ）の各要素の値を更新している。その後、文２５～２９で、配列インデックスＩが１から９９の範囲について、配列Ｃ（Ｉ）を更新し（文２６）、続いて配列要素Ｂ（Ｉ）の値を更新する処理（文２７）を繰り返している。次に、文２９で変数ＮＴに１を加えている。

【００２３】プログラム並列化処理とは、このような単一プロセッサ用のプログラムから、複数のプロセッサで並列に実行するようなプログラムを生成する処理のことである。例えば、図３に示す元のプログラム２００では、配列Ａ，Ｂ，Ｃが用いられているので、これらの配列を適宜複数の範囲に分割して、各範囲の計算を各プロセッサに割り当てる。すなわち、各プロセッサが、配列Ａ，Ｂ，Ｃの更新処理（文１１，２３，２６，２７）のうち、配列インデックスが＄ｌｂから＄ｕｂまでの範囲（ただし、＄ｌｂ、＄ｕｂは、各プロセッサ毎に異なる値が設定される変数を表すものとする）を更新する処理を分担することにより、複数のプロセッサが並列に処理を実行するような並列化プログラムを生成する。

【００２４】初めに、従来技術によるプログラム並列化処理によって生成されるプログラムを説明する。図４は、図３の元のプログラムに対して従来技術によるプログラム並列化処理を施すことによって生成した並列化プログラム５００を示す。プログラム５００は、複数あるプロセッサの各々で実行されるプログラムである。ただし、上述したように変数＄ｌｂ、＄ｕｂの値は各プロセッサごとに異なる。

【００２５】図３に示す元のプログラム２００において、配列Ａ，Ｂ，Ｃを更新する文１１，２３，２６，２７の処理は分割されて、各プロセッサが分担することになる。このために、図４に示す並列化プログラム５００では、元のプログラム２００のＤＯ文１０，２２，２５のループ範囲を変更し、それぞれ文１０－ａ，２２－ａ，２５－ａに置き換えている。

【００２６】図４に示すプログラム５００において、条件文２１の実行では、すべてのプロセッサが配列要素Ｂ（１）の値を参照する。このため、プログラム並列化処理は、配列要素Ｂ（１）を所有しているプロセッサから他のすべてのプロセッサに配列要素Ｂ（１）の値を転送する通信コード（文２０１，２０２）を生成する。また、文２７の実行において、各プロセッサは自己が所有していない配列要素Ａ（＄ｕｂ＋１）の値を参照する。このため、プログラム並列化処理は、この値の転送を行う通信コード（文２４２，２４３）を生成する。

【００２７】ここで、図４のプログラム５００において、配列Ａの値を更新する文２３は、条件文２１の評価結果に応じて実行するか否かが決定される。条件文２１の評価結果により文２３が実行されなかった場合、配列Ａの値は更新されていない。従って、文２７で参照するＡ（Ｉ＋１）の値は、ＤＯ文２０の繰り返しにおけるそれ以前の文２７の実行の際に参照した値のままであるから、このときのプロセッサ間通信（文２４２，２４３）は本来は不要であるということになる。

【００２８】しかし、従来の並列化処理では、その要否はプログラム並列化を行う時点では判定できない。このため、従来方式では、図４に示すようにその要否に関係なく通信を行うような並列計算機用のプログラムを生成してしまう。

【００２９】次に、本実施例の方式によるプログラム並列化処理によって生成したプログラムについて説明する。図５は、図３の元のプログラム２００を用いて、本実施例によるプログラム並列化処理（図１および２）を施すことによって生成した並列化プログラム５０１を示す。

【００３０】図５のプログラム５０１では、文２２－ａおよび文２３からなるブロックが実行されたか否かを、フラグＡ＿ｕｐｄａｔｅ＿ｆｌａｇの値によって管理している。すなわち、文２２－ａおよび文２３からなるブロックを実行したときには、文２４０でフラグＡ＿ｕｐｄａｔｅ＿ｆｌａｇの値を１に更新する。そして、文２５－ａから文２８までのブロックを実行する前に、フラグＡ＿ｕｐｄａｔｅ＿ｆｌａｇが１の場合には、プロセッサ間通信（文２４２，２４３）を実行し、フラグＡ＿ｕｐｄａｔｅ＿ｆｌａｇが０の場合には、プロセッサ間通信（文２４２，２４３）を実行しないように制御する

条件文２４１，２４５が挿入されている。これにより、本実施例によって並列化したプログラム５０１では、無駄なプロセッサ間通信を行うことがない。

【００３１】本来は、図５のプログラム５０１では、文２６および２７から成るブロック（図１０の基本ブロック４）で、プロセッサ間通信をおこなった後の配列Ａをあるプロセッサが用いる必要があるか否かは、そのプロセッサが参照すべき配列値を所有する他のプロセッサが文２３のブロック（図１０の基本ブロック３）を実行したかどうかで判断しなければならない。しかし、本発明によるプログラム並列化方法では、並列実行するプロセッサは、それぞれ並列化プログラム中の基本ブロックを同じ順序で実行するという事実に基づいて実現されている。このため、他のプロセッサが基本ブロック３を実行したかどうかは、自分自身のプロセッサが基本ブロック３を実行したかどうかに置き換えることによって知ることができる。自分自身が基本ブロック３を実行したかどうかは、基本ブロック３の最後でフラグ変数をセットするような実行文を挿入することにより、該フラグ変数の値で判定できる。これにより、前記のように並列化プログラムの実行時にプロセッサ間通信の要否を判定することができる。

【００３２】次に、このような無駄な通信を行わない並列化されたプログラムを自動生成する本実施例の処理手順を詳細に説明する。

【００３３】図１を参照して、本実施例のプログラム並列化処理の処理手順を説明する。計算機１は、まずプログラム２を入力し、プログラム分割処理３を行う。これは、データの分割処理３１および処理の分割３２の決定、すなわち入力したプログラム２の処理およびデータをどのように分割して複数のプロセッサに割り当てるかを決定する処理である。データ分割３１は図６に示すような、プログラム中の各配列をプロセッサにどう分割して割り当てるかを示す表を作成する。処理の分割３２は、プログラム中の各実行文をどう分割してどのプロセッサが実行するかを決定し、例えば図７に示すような、各DOループをどのようにプロセッサが分担して実行するかを示す並列実行制御情報８を生成し、さらに該並列実行制御情報８を参照して各プロセッサが実行する共通の並列化されたソースプログラム７（図９）を生成してそれぞれファイルに格納する。処理およびデータをどのように分割するかを決定する方式は、任意でよい。続いて、通信コード生成処理４を実行して、並列化プログラム５を生成する。

【００３４】該生成された並列化プログラム５を実行する並列プロセッサの構成を図８に示す。並列プロセッサは以下のような機能をもつ。

【００３５】（１）複数のプロセッサ４０ａ～４０ｄとそれらプロセッサ間のデータ転送を可能とするネットワーク４２とを有する。

【００３６】（２）プロセッサ４０ａ～４０ｄは、それぞれデータやプログラムを保持するためのメモリ４１ａ～４１ｄを有する。

【００３７】（３）各プロセッサは他のプロセッサとの間で通信ネットワーク４２を介してデータを送受信する機能を持つ。データの送受信は、以下の２つのシステムコールによって実行される。

【００３８】(3.1) データ送信処理は、sendシステムコールにより実行される。sendシステムコールは、送信対象変数、送信先プロセッサ番号及びデータ識別子を引数として実行される。

【００３９】(3.2) データ受信処理は、receiveシステムコールにより実行される。receiveシステムコールは、受信データを代入する変数及びデータ識別子を引数として実行される。receiveシステムコールを実行したとき識別子に対応するデータが到着していない場合は、プロセッサはデータが到着するまで、他の処理の実行を休止して待つ。

【００４０】次に、図２のチャートを参照して、通信コード生成処理４について詳細に説明する。ここでは、図９に示す並列化プログラム（通信コードなし）７を元に通信コードを付加する場合を説明する。しかし、図１の破線の矢印に示すように、図３に示す元のプログラム２００の分割処理３をおこなった後、そのまま通信コード生成処理をおこなうようにしても良い。

【００４１】図２において、通信コード生成処理４では、先ずプログラムの制御フロー解析４０１を行い、プログラムの実行の流れを有向グラフで表現した制御フローグラフ４１１を作成する。図１０は、図９の並列化ソースプログラム７に対して制御フロー解析４０１を行うことにより得られた制御フローグラフ４１１の概念図を、また、図１１は、該制御フローグラフ４１１のデータ構造をそれぞれ示す。図１０において、４１００～４１０３は、逐次的に実行される一連の代入文や式の列からなる基本ブロックを示す。矢印４１１０～４１１６は、基本ブロック間の制御の流れを表す枝を示す。

【００４２】次に、計算機１は、図２におけるデータ依存解析４０２を行い、プログラム中の変数の定義・参照の関係を有向グラフで表現したデータ依存グラフ４１２を作成する。図１２は、図３に示す元のプログラム２００に対するデータ依存グラフ４１２の概念図を、また、図１３は該データ依存グラフ４１２のデータ構造をそれぞれ示す。矢印４１２０～４１２４は、データの定義・参照の関係を示す。各矢印の起点はそのデータが定義されているところを示し、終点はそのデータが参照されているところを示している。すなわち、図１２に示すデータ依存解析により得られる図１３の表４１２から、各配列について、その配列を定義するブロックおよびその配列を参照するブロックのすべてを知ることができる。具体的には、図１３において、例えば配列Ａは、ブロック

４１０２（基本ブロック３）によって定義され、ブロック４１０３（基本ブロック４）の２カ所で参照されていることがわかる。

【００４３】次に、図２における通信データ抽出処理４０３を行う。これは、データ依存グラフ４１２に現れる各矢印、すなわちデータ依存グラフ中のそれぞれのデータの定義・参照の関係について、プロセッサ間で通信をおこなう可能性のあるものを選びだし、通信対象データ依存グラフ４１３を作成する処理である。通信対象データ依存グラフ４１３は、データ依存グラフ４１２の部分グラフとなる。

【００４４】図１４は、図１２のデータ依存グラフ４１２から作成した通信対象データ依存グラフ４１３の概念図を、図１５は該通信対象データ依存グラフ４１３のデータ構造をそれぞれ示す。図１２のデータ依存グラフ４１２の中の矢印４１２１，４１２３については、そのデータを定義するプロセッサと参照するプロセッサとが同一であることが分かる。従って、これらの矢印４１２１，４１２３は、プロセッサ間通信が必要ないので、取り除かれる。一方、図１２の矢印４１２０、４１２２および４１２４は、プロセッサ間通信が必要なものとして取り出される。その結果、図１４に示す通信対象データ依存グラフ４１３が得られる。

【００４５】次に、図２における通信位置決定処理４０４を行う。この処理により、通信データ抽出処理４０３で抽出されたプロセッサ間通信をおこなう可能性のあるデータ依存について、並列化プログラム中でプロセッサ間通信を行う箇所を決定し、その位置に、必要なプロセッサ間通信のための通信コード（通信情報）を挿入する。本実施例では、プロセッサ間通信対象のデータを参照する基本ブロックの直前で通信を行うように、挿入位置を決定する。

【００４６】図１６は、図１４の通信対象データ依存グラフ４１３から得られた通信情報リスト４１４を、図１７は該通信情報リスト４１４のデータ構造をそれぞれ示す。４１４１は、図１４の通信対象データ依存グラフ４１３中のデータ依存４１２０および４１２４に基づいて生成した通信情報である。通信情報４１４１は、基本ブロック４１０１の直前に配置される。この通信情報４１４１は、自プロセッサが配列要素Ｂ（１）を所有しているときは他のすべてのプロセッサにその値を送信し、自プロセッサが配列要素Ｂ（１）を所有していないときはＢ（１）を所有するプロセッサからその値を受信するという処理を行うものである。

【００４７】図１６に示す通信情報４１４２は、図１４の通信対象データ依存グラフ４１３中のデータ依存４１２２に基づいて生成した通信情報である。通信情報４１４２は、配列要素Ａ（＄ｌｂ）を自プロセッサの直前のプロセッサ（プロセッサのＩＤが自プロセッサより１小さいもの）に送信し、配列要素Ａ（＄ｕｂ＋１）を自プロセッサの直後のプロセッサ（プロセッサのＩＤが自プロセッサより１大きいもの）から受信するという処理を行うものである。

【００４８】次に、計算機１は、図２に示す通信条件決定処理４０５を行なう。これは、図２の通信位置決定処理４０４で挿入した各通信情報（図１６の４１４１や４１４２）について、データ依存グラフ４１２（図１２）と制御フローグラフ４１１（図１０）とから実際に通信が必要になる条件を求め、通信の実行を制御するための制御情報を作成する処理である。通信が必要になる条件は、通信対象データ依存グラフ４１３（図１４）に現れる矢印のデータ依存が実行中に実際に生ずる条件であり、これは制御フローグラフ４１１（図１０）を追跡することによって求めることができる。

【００４９】図１８は、図２に示す通信条件決定処理４０５の詳細なチャートを示す。図１９は、図１８の通信条件決定処理４０５の中間段階の情報４０５５を示す。以下、図１８に示したチャート、および図１９の情報４０５５を参照して、通信条件決定処理４０５の処理を説明する。

【００５０】通信条件決定処理４０５は、図１６のような通信情報リスト４１４の各通信情報について以下の処理を行う。

【００５１】図１８において、まず、変数定義フラグ生成処理４０５１は、プログラム実行中に通信対象の変数値の定義が行われたか否かを表すフラグ変数を設定するコード情報を作成する。次に、通信条件式生成処理４０５２は、該通信の要否を実行時に判定するための条件式を作成する。

【００５２】さらに、変数定義フラグ初期化コード生成処理４０５３は、通信を行った場合に変数定義フラグを初期化するコード情報を作成する。また、通信条件簡約化処理４０５４は、通信条件式生成処理４０５２で作成した条件式から、制御フローグラフ４１１を参照して自明な条件を取り除いて簡約化する。

【００５３】具体的に説明すると、図１６の通信情報リスト４１４の通信情報４１４１については、この通信がデータ依存４１２０および４１２４（図１４）から生じていることから、基本ブロック４１００または４１０３を通った場合に通信対象データＢ（１）の定義が起こることが分かる。そこで、変数定義フラグ生成処理４０５１は、フラグＢ__ｕｐｄａｔｅ__ｆｌａｇを設け、基本ブロック４１００または４１０３を通った場合にこの値を１に設定するコードの情報４１５４，４１５７（図１９参照）を作成する。

【００５４】次に、プロセッサ間通信はフラグＢ__ｕｐｄａｔｅ__ｆｌａｇが１になっている場合に限って実行する必要があることから、図１８の通信条件式生成処理４０５２は、条件式４１５５を作成する。次に、変数定義フラグ初期化情報生成処理４０５３は、通信実行後に

フラグB__update__flagの値を0に初期化するコード情報4156を作成する。

【0055】さらに、通信条件簡約化処理4054は、条件式で参照するフラグB__update__flagについて、制御フローグラフ411（図10）を逆に辿って、その値を定義する式を調べる。この例では、制御フローグラフ411の経路4110と4114を逆に辿ると、共に、フラグB__update__flagは1になっていることが分かる。このことは、図11に示す制御フローグラフ411のデータ構造において、ブロック2の先行ブロックが1と4のみであることからも分かる。このため、条件式4155は常に成立することが確定するので、条件式は不要であると判定できる。そこで、条件式4155およびこれに用いられる変数定義フラグB__update__flagを削除する。

【0056】図16の通信情報4142についても同様にして、フラグA__update__flagを設け、基本ブロック4102を通った場合にこの値を1に設定するコードの情報4151（図19）を設ける。また、条件式4152、およびフラグA__update__flagを初期化するコード情報4153を作成する。

【0057】次に、制御フローグラフ411（図10）を逆に辿ってフラグA__update__flagの値を定義する式を調べる。この場合は、経路4112および4113を調べる。経路4113を逆に辿った場合は、フラグA__update__flagの値が1になっていることが分かる。しかし、経路4112を逆に辿った場合は、フラグA__update__flagの値は確定できない。このことは、図11におけるブロック4の先行ブロックが2、3および4であるのに、図15における配列Aについての参照ブロック4の定義ブロックが3のみであることからも分かる。このため条件式4152などはそのまま残す。

【0058】以上のようにして、図16の通信情報リスト414から図20に示すような通信制御情報リスト415が生成される。図20の通信制御情報リスト415では、フラグA__update__flagを用いた条件文4152によって、並列化プログラムの実行時に、各プロセッサが、通信コード4142の実行の要否を判断するようになっている。従って、不要な場合はプロセッサ間通信を行なわないので、通信の回数や負荷が軽減される。このような条件文はすべての通信コードに付加されるわけではなく、通信コード4141のように常に必要であると判断される通信コードでは（条件文が付加されずに）無条件にプロセッサ間通信を行なうようになっている。従って、無駄に条件文を実行することもない。図21は通信制御情報リスト415のデータ構造を示す。

【0059】再び図2を参照して、コード生成処理460は、ここまでに決定した処理の分割、通信情報に基づいて、図5に示したような並列プロセッサ用の並列化プログラムを出力する。

【0060】以上説明した実施例では、配列は1次元であった。したがって、参照フラグの値は、プロセッサ間通信をおこなうか否かの2値でよかった。しかし、配列が2次元以上の場合、ある配列要素に隣接する配列要素は、4値以上となるため、参照フラグは、"1"か"0"かの2値では不十分となる。

【0061】この問題に対処した、本発明の別の実施例について説明する。本実施例は、1つの配列を複数回参照し、かつ参照する配列要素インデックスが異なるような場合について、本発明によるプロセッサ間通信コード生成処理を適用したものである。この場合には、プロセッサ間通信の要否を配列毎に判定する方法では無駄な通信を十分に取り除くことはできず、1つの配列について通信対象部分を分割して、それぞれについてプロセッサ間通信の要否を判定する必要が生ずる。以下では本実施例による通信コード生成処理を、図22に示す通信コードなしの並列化プログラム310に適用した場合について説明する。

【0062】プログラム310は、ブロック311で配列Aを更新し、該更新後の値をブロック312およびブロック313で参照する。ブロック312は条件ブロックであり、条件文（文324）の判定条件が成立した場合にのみ実行される。配列Aはプログラム310中の2つのブロックで参照されるが、各々の参照する配列要素インデックスは異なる。すなわち、ブロック312ではB（I，J）の更新のためにA（I－1，J）、A（I＋1，J）を参照し、ブロック313ではA（I－1，J）、A（I＋1，J）、A（I，J－1）、A（I，J＋1）を参照する。

【0063】ここで、2次元配列のこのようなデータ参照の仕方をデータ参照パターンとして図23のようにグラフで表現することにする。図23で、例えばSTENCIL1はA（I－1，J）、A（I＋1，J）、A（I，J－1）、A（I，J＋1）の4点を参照することを示す参照パターンである。同様に、STENCIL2は、A（I－1，J）、A（I＋1，J）の2点を参照する参照パターン、STENCIL3はA（I，J－1）、A（I，J＋1）の2点を参照する参照パターンである。この参照パターンを用いて表せば、プログラム310では、ブロック312で参照パターンSTENCIL2、ブロック313で参照パターンSTENCIL1で配列Aを参照する。プロセッサ間で通信すべきデータはこの参照パターンによって決定できる。このような参照パターンは参照点の相対的なインデックスを要素とする配列として表現できる。該参照パターンを表す配列と通信対象の配列名とを引数として必要なデータの送信、受信を行う関数をそれぞれsendp, recvpとする。

【0064】以下では、まず本実施例によるプログラム

並列化方式をプログラム３１０の例で説明し、次にこのような並列化プログラムを生成する方法を説明する。

【００６５】プログラム３１０を本実施例によって並列化したプログラムを図２４に示す。プログラム中、配列Ａに関係する参照パターンは、図２３のSTENCIL1, STENCIL2の２通りである。配列のうち、各プロセッサが参照のみ行い、更新処理を行わない部分を参照エリアと呼ぶ。各プロセッサは参照エリアのデータを参照する場合、プロセッサ間通信によって他のプロセッサからデータを受け取る必要がある。変数A_status_flagは、配列Ａの参照エリアのうち最新の値に更新されていない（以下これをDirtyであるという）部分に対応する参照パターン番号を保持する。配列Ａが更新された直後には参照エリアはすべてDirtyとなり、Dirtyな部分はSTENCIL1に相当する。この場合、プログラムではA_status_flagを1にする。ブロック３１２の配列Ａの参照パターンはSTENCIL2なので、これに対応するプロセッサ間通信を行う（文３２４１、３２４２）。

【００６６】このプロセッサ間通信によって、配列Ａの参照エリアのうち一部分は通信済みとなり、STENCIL3に相当する部分のみDirtyとなる。これを示すために制御変数A_status_flagの値を３とする（文３２４３）。ブロック３１３の配列Ａの参照パターンはSTENCIL1であるが、ブロック３１３実行開始時点では、ブロック３１２を実行してSTENCIL2を参照パターンとするプロセッサ間通信を実行済みの場合とそうでない場合の２通りがある。ブロック３１２を実行していない場合は、参照パターンSTENCIL1でのプロセッサ間通信が必要である。一方、ブロック３１２を実行済みの場合は、STENCIL1とSTENCIL2の差分として参照パターンSTENCIL3でのプロセッサ間通信が必要である。これを、制御変数A_update_flagの値で判定し、通信を実行する（文３２９１～３２９７）。

【００６７】通信コード生成処理４はプロセッサ間データ依存がある各基本ブロックについて、該基本ブロック実行時に通信が必要なデータを判定し、必要なデータだけを通信するようなプログラムを生成する。図２の制御フロー解析４０１から通信位置決定処理４０４までは、最初の実施例の場合と全く同様に行う。

【００６８】図２の通信条件決定処理４０５及びコード生成処理４０６は、初めに通信対象となる各配列についてプログラム中で現れる参照パターン、及びそれらの参照パターンの和、交わりをとることによって得られる参照パターンをすべて列挙し、参照パターンに通し番号を付ける。ここで、参照パターンの和、差とは、参照点の集合としての和、差を意味する。また、すべての参照パターンの和をとって得られる参照パターンをＵとする。プログラム３１０の例では、図２３に示す３通りの参照パターンを得る。

【００６９】次に通信情報リストの各要素について、状態管理変数の示す参照パターンＳ、該ブロックでのデータ参照パターンＲに対して、参照パターンＲ－（Ｕ－Ｓ）でプロセッサ間通信を行う実行文列（図５、文２４１～２４２、２９１～２９７）を生成する。次に、状態管理変数を参照パターンＳ－Ｒの番号に設定する実行文列（図５、文２４３、２９８）を生成する。ここで減算記号は、集合としての差分を意味する。これにより状態管理変数は、該配列の通信エリアのうち、更新されていない部分に相当する参照パターンの番号を常に保持し、また各通信において本当に必要な部分だけを通信するように動作する。

【００７０】以上のようにして、図２４に示すような並列化プログラムが生成される。

【００７１】

【発明の効果】本発明のプログラム並列化方法によれば、並列実行時の通信の回数を従来方法に比べて削減することができ、従って通信に要する実行負荷を削減することができる。これにより生成した並列計算機用プログラムの実行性能を向上させることができる。

【図面の簡単な説明】

【図１】本発明のプログラム並列化方法を実現する計算機システムの一実施例の概略構成および概略処理を示す図。

【図２】図１に示す計算機システムによる通信コード生成処理の例を示すチャート。

【図３】図３、単一プロセッサ用のプログラムの例を示すプログラムリスト。

【図４】図３の単一プロセッサ用のプログラムを従来のプログラム並列化方法により並列化したプログラムリスト。

【図５】図３の単一プロセッサ用のプログラムを本発明のプログラム並列化方法により並列化したプログラムリスト。

【図６】図１に示すデータ分割情報６の例を示す表。

【図７】図７は、図１に示す並列実行用制御情報８の例を示す表。

【図８】図８は、本発明により生成された並列化プログラムを実行する並列プロセッサシステムの構成例を示す概略図。

【図９】図１に示す並列化プログラム（通信コードなし）７の例を示す図。

【図１０】図２に示す制御フローグラフ４１１の例を示す図。

【図１１】図１０に示す制御フローグラフ４１１のデータ構造を示す表。通信条件決定処理フローチャート。

【図１２】図２に示すデータ依存グラフ４１２の例を示す図。

【図１３】図１２に示すデータ依存グラフ４１２のデータ構造を示す表。

【図１４】図１４は、図２に示す通信対象データ依存グ

17

ラフ４１３の例を示す図。

【図１５】図１４に示す通信対象データ依存グラフ４１３のデータ構造を示す表。図１３の単一プロセッサ用のプログラムを本実施例により並列化したプログラム。

【図１６】図２に示す通信制御情報リスト４１４の例を示す図。

【図１７】図１６に示す通信制御情報リスト４１４のデータ構造を示す表。

【図１８】図２に示す通信条件決定処理４０５の詳細な処理を示すチャート。

【図１９】図１８に示す通信条件決定処理４０５の中間段階における情報の例を示す図。

【図２０】図２に示す通信制御情報リスト４１５の例を示す図。

【図２１】図２０に示す通信制御情報リスト４１５のデータ構造を示す表。

【図２２】２次元配列を含む単一プロセッサ用のプログラムの例を示すプログラムリスト。

18

【図２３】２次元配列の参照パターンの例を示す説明図。

【図２４】図２２に示す単一プロセッサ用のプログラムを本発明の実施例により並列化したプログラムリスト。

【符号の説明】

１・・・プログラム並列化処理、２・・・ＦＯＲＴＲＡＮプログラム

３・・・プログラム分割処理、４・・・通信コード生成処理

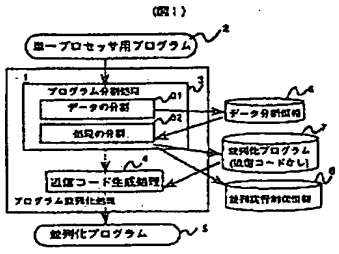
５・・・並列計算機用プログラム、４０３・・・通信データ抽出処理

４０４・・・通信位置決定処理、４０５・・・通信条件決定処理

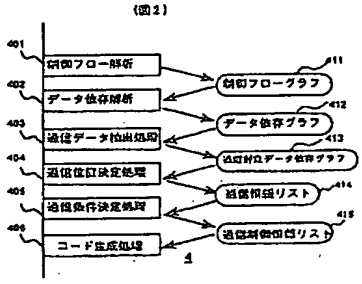
４１１・・・制御フローグラフ、４１２・・・データ依存グラフ

４１３・・・通信通信対象データ依存グラフ、４１４・・・通信情報リスト、４１５・・・通信制御情報リスト。

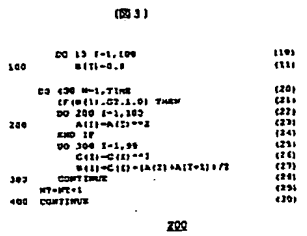
【図１】



【図２】



【図３】

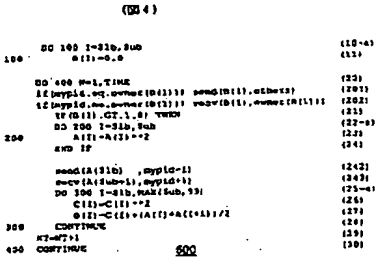


```
      DO 15 I=1,100                                    (10)
100     B(I)=0.0                                       (11)

      DO 400 M=1,TIME                                  (20)
        IF(W(I).GT.1.0) THEN                           (21)
        DO 200 I=1,100                                 (22)
          A(I)=A(I)**2                                 (23)
        END IF                                         (24)
        DO 300 I=1,99                                  (25)
          C(I)=C(I)**2                                 (26)
          B(I)=C(I)*(A(I)+A(I+1))/2                    (27)
300     CONTINUE                                       (28)
      MT=MT+1                                          (29)
400   CONTINUE                                         (30)
```

200

【図４】



```
      DO 105 I=$lb,$ub                                 (10-4)
105     B(I)=0.0                                       (11)

      DO 400 M=1,TIME                                  (20)
      IF(myrid.eq.owner(B(1))) send(B(1),other)        (201)
      IF(myrid.eq.owner(B(1))) recv(B(1),owner(B(1)))  (202)
        IF(B(1).GT.1.0) THEN                           (21)
        DO 200 I=$lb,$ub                               (22-8)
          A(I)=A(I)**2                                 (23)
        END IF                                         (24)

        send(A($lb)   ,myrid-1)                        (242)
        recv(A($ub+1),myrid+1)                         (243)
        DO 300 I=$lb,MAX($ub,99)                       (25-4)
          C(I)=C(I)**2                                 (26)
          B(I)=C(I)*(A(I)+A(I+1))/2                    (27)
300     CONTINUE                                       (28)
      MT=MT+1                                          (29)
400   CONTINUE                                         (30)
```

600

【図5】

(図5)

```
      DO 100 I=Slb,Sub                                    (10-a)
100       B(I)=0.0                                        (11)

      DO 400 N=1,TIME                                     (20)
      IF(mypid.eq.owner(A(1))) send(B(1),others)          (201)
      IF(myid.NE.owner(B(1))) recv(B(1),owner(B(1)))      (202)
        IF(B(1).GT.1.0) THEN                              (21)
        DO 200 I=Slb,Sub                                  (22-a)
200         A(I)=A(I)**2                                  (23)
        A_update_flag=1                                   (240)
        END IF                                            (24)


      IF(A_update_flag .eq. 1) then                       (241)
         send(A(Slb)  ,mypid-1)                           (242)
         recv(A(Sub+1),mypid+1)                           (243)
         A_update_flag=0                                  (244)
      end if                                              (245)
        DO 300 I=Slb,MAX(Sub,99)                          (25-a)
          C(I)=C(I)**2                                    (26)
          B(I)=C(I)+(A(I)+A(I+1))/2                       (27)
300     CONTINUE                                          (28)
        NT=NT+1                        501                (29)
400   CONTINUE                                            (30)
```

【図6】

(図6)

8

変数・配列名	プロセッサ0		プロセッサ1		プロセッサ2		プロセッサ3	
	上限	下限	上限	下限	上限	下限	上限	下限
A	1	25	26	50	51	75	76	100
B	1	25	26	50	51	75	76	100
C	1	25	26	50	51	75	76	100

【図7】

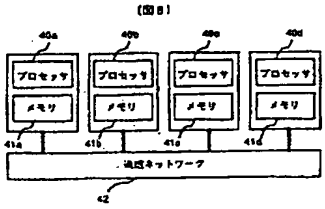
(図7)

8

DO文番号		プロセッサ0		プロセッサ1		プロセッサ2		プロセッサ3	
		上限	下限	上限	下限	上限	下限	上限	下限
200	局所変数名	Slb	Sub	Slb	Sub	Slb	Sub	Slb	Sub
	値	1	25	26	50	51	75	76	100
300	局所変数名	Slb	Sub	Slb	Sub	Slb	Sub	Slb	Sub
	値	1	25	26	50	51	75	76	100

【図8】

(図8)



【図9】

(図9)

Z

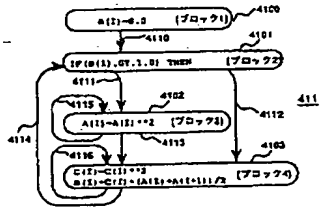
```
      DO 100 I=Slb,Sub                                    (10-a)
100       B(I)=0.0                                        (11)

      DO 400 N=1,TIME                                     (20)
        IF(B(1).GT.1.0) THEN                              (21)
        DO 200 I=Slb,Sub                                  (22-a)
200         A(I)=A(I)**2                                  (23)
        END IF                                            (24)

        DO 300 I=Slb,MAX(Sub,99)                          (25-a)
          C(I)=C(I)**2                                    (26)
          B(I)=C(I)+(A(I)+A(I+1))/2                       (27)
300     CONTINUE                                          (28)
        NT=NT+1                                           (29)
400   CONTINUE                                            (30)
```

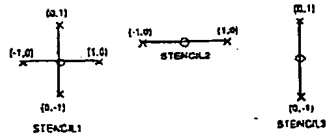
【図10】

(図10)



【図23】

(図23)



【図１１】

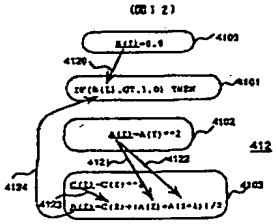
(図１１)

411

ブロック	先行ブロック	行き先ブロック
1	φ	2
2	1,4	3,4
3	2,3	3,4
4	2,3,4	2,4

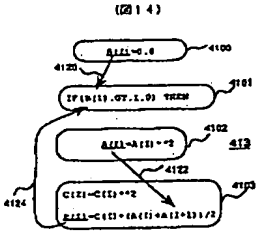
【図１２】

(図１２)



【図１４】

(図１４)



【図１３】

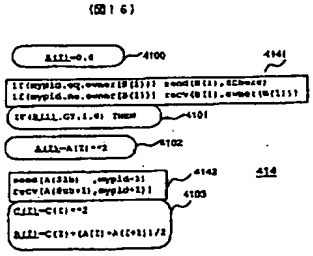
(図１３)

412

変数・配列名	定義ブロック	参照ブロック	参照インデックス	
			相対/絶対	インデックス
B	1	2	絶対	1
A	3	4	絶対	0
A	3	4	相対	+1
C	4	4	相対	0
B	4	2	絶対	1

【図１６】

(図１６)



【図１５】

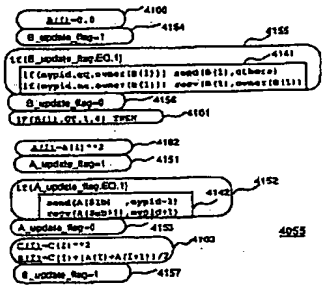
(図１５)

413

変数・配列名	定義ブロック	参照ブロック	参照インデックス	
			相対/絶対	インデックス
B	1	2	絶対	1
A	3	4	相対	+1
B	4	2	絶対	1

【図１９】

(図１９)



【図１７】

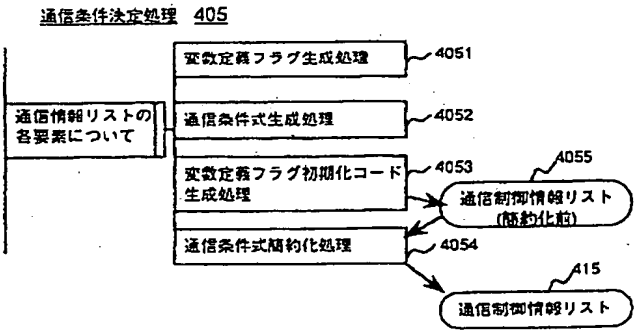
(図１７)

414

変数・配列名	定義ブロック	参照ブロック	通信対象インデックス		送信元	受信先
			相対/絶対	インデックス		
B	1 4	2	絶対	1	B(1)を定義するプロセッサ	全
A	3	4	相対	+1	全	全

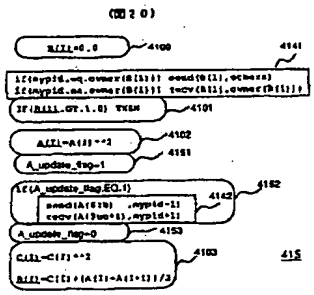
【図１８】

(図１８)



【図２０】

(図２０)



【図２１】

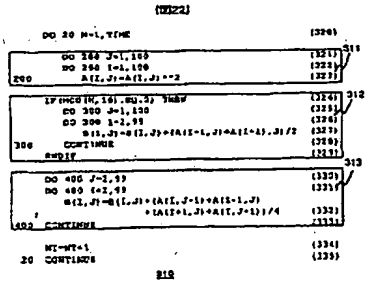
(図２１)

415

変数・配列名	通信ブロック	通信対象インデックス 絶対/絶対	通信対象インデックス インデックス	送信元	受信先	通信実行条件
B	2	絶対	1	既(1)の所有者	全	常に
A	4	相対	+1	全	全	A_update_flag=1

【図２２】

(図２２)



【図２４】

(図２４)

```
      DO 20 M=1,TIME                                          (3320)

        DO 220 J=1,100                                        (3321)
        DO 210 I=1,100                                        (3322)
210       A(I,J)=A(I,J)**2                                    (3323)
        A_update_flag=1                                       (3324)

      IF(MOD(M,10).EQ.0) THEN                                 (3325)
        send(A,SYNC(1))                                       (3326)
        recv(A,SYNC(1))                                       (3327)
        A_update_flag=2                                       (3328)
        DO 300 J=1,100                                        (3329)
        DO 300 I=2,99                                         (3330)
          B(I,J)=B(I,J)+(A(I-1,J)+A(I+1,J))/2                 (3331)
300     CONTINUE                                              (3332)
      ENDIF                                                   (3333)

      if(A_update_flag .eq. 1) then                           (3334)
        send(A,SYNC(1))                                       (3335)
        recv(A,SYNC(1))                                       (3336)
      else if(A_update_flag .eq. 2) then                      (3337)
        send(A,SYNC(2))                                       (3338)
        recv(A,SYNC(2))                                       (3339)
      end if                                                  (3340)
      A_update_flag=0                                         (3341)
      DO 400 J=2,99                                           (3342)
      DO 400 I=2,99                                           (3343)
        B(I,J)=B(I,J)+(A(I,J-1)+A(I-1,J)
     &              +(A(I+1,J)+A(I,J+1))/4                    (3344)
400   CONTINUE                                                (3345)

      MT=MT+1                                                 (3346)
20    CONTINUE                                                (3347)
```

25　　26

フロントページの続き

(72)発明者　金野　千里
東京都国分寺市東恋ケ窪１丁目280番地
株式会社日立製作所中央研究所内

(72)発明者　猪貝　光祥
東京都小平市上水本町５丁目20番１号　日立超エル・エス・アイ・エンジニアリング株式会社内